Fine-grained management of CoAP interactions with constrained IoT devices

Floris Van den Abeele Jeroen Hoebeke Ingrid Moerman Piet Demeester
Department of Information Technology, Ghent University – iMinds
Gaston Crommenlaan 8/201 B-9050 Ghent, Belgium
floris.vandenabeele@intec.ugent.be

***Abstract*—As open standards for the Internet of Things gain traction, the current Intranet of Things will evolve to a truly open Internet of Things, where constrained devices are first class citizens of the public Internet. However, the large amount of control over constrained networks offered by today's vertically integrated platforms, becomes even more important in an open IoT considering its promise of direct end-to-end interactions with constrained devices. In this paper a set of challenges is identified for controlling interactions with constrained networks that arise due to their constrained nature and their integration with the public Internet. Furthermore, a number of solutions are presented for overcoming these challenges by means of an intercepting intermediary at the edge of the constrained network.**

I. Introduction

In the current Internet of things many vendors rely on their own platforms to create vertical solutions, a phenomenon that is commonly referred to as the Intranet of Things [1]. The resulting products are typically only accessible to the manufacturer. While this approach gives manufacturers a maximum of control over their product, it also severely hampers re-using existing things and technology in new services. This in turn restricts the opportunities for new service providers to enter the IoT market.

Standardization is one piece of the answer to break open the Intranet of things. There have been numerous standardization efforts for the IoT by standard developing organization (SDOs) like the W3C, ETSI, OGC and many others. The SDO most applicable to the Internet - the Internet Engineering Task Force (IETF) - however, is developing a protocol stack [2] for integrating constrained node networks [3] (CNNs) into the public Internet (see Figure 1). Most notable is the recent

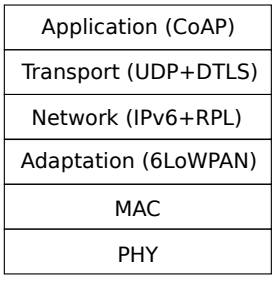

Fig. 1: IETF IoT protocol stack

ratification of an application-layer protocol named the Constrained Application Protocol (CoAP) [4]. CoAP employs a RESTful client-server design for interacting with constrained devices. Devices essentially become web services that can be consumed by applications, thus giving rise to a so-called Web of Things [5] (WoT). CoAP relies on transport layer security in the form of DTLS, which is in simple terms TLS over UDP. The IETF DICE working group is looking to standardize a set of DTLS protocol parameters for use in the IoT domain. Apart from an application-layer protocol the IETF has also standardized 6LoWPAN, a protocol which facilitates the use of IPv6 inside CNNs, and RPL, a routing protocol for CNNs. Next to communicating application data, CoAP can also be re-used as a protocol to manage devices (e.g. access control lists, device settings and other parameters), as is the case in the Open Mobile Alliance lightweight M2M standard [6].

The availability of open standards alone is not enough to open up the Intranet of Things. As mentioned, one of the benefits of employing vertical silos is the strict control of owners over their devices and data. This control is necessary due to the constrainedness (in terms of processing power and energy budget) of the devices and the networks involved. A network running only the IETF IoT protocol lacks the level of control typically found in vertically integrated products. Clearly, in order to expedite the adoption of these open standards the needs of vendors for controlling and managing constrained networks and devices should be addressed.

Our contribution in this paper consists of identifying a number of open problems for controlling interactions with constrained networks and nodes that are running the open IETF IoT protocol stack on so-called class 1 devices. Capabilities of class 1 devices are limited to $\sim$10 KiB of RAM and $\sim$100 KiB of ROM [3]. Furthermore, we propose a flexible architecture that enables chaining together function blocks - known as adapters - on an intermediary system to build specific solutions for the identified problems. This architecture enables us to implement fine-grained and configurable control and management of constrained devices.

II. Issues with running the open IETF IoT stack

While naively deploying the IETF IoT stack on class 1 devices would effectively tear down the Intranet of Things, the resulting direct end-to-end connectivity with any other Internet host would also cause a number of serious threats for the constrained network. When left unaddressed, these threats could severely compromise the network's operation. As will be shown, constrained devices themselves are often unsuited to address these issues due to their limited resources and available energy budget. This section categorizes these threats. Section III outlines our approach for tacking these issues. This leads to a number solutions in section IV.

A. Security with DTLS

In the IETF stack of Figure 1, the preferred transport layer security protocol is DTLS. While DTLS offers endpoint authentication, data integrity and confidentiality; it does so at considerable cost for class 1 devices.

Firstly, DTLS is difficult to deploy on this type of devices. As mentioned in [7], the code footprint of a DTLS implementation on a constrained device lies around 16 KB of ROM and 4 KB of RAM. Considering the capabilities of class 1 devices mentioned in the previous section, the large RAM footprint is problematic. Since the DICE working group did not reach consensus to change the workings of the DTLS protocol itself, this footprint is expected to stay unchanged in the near future.

Apart from the footprint, communication overhead is a second issue for all types of constrained devices. This overhead is primarily caused by the DTLS handshake, which has to occur between any two hosts that want to setup a DTLS session. The overhead becomes unworkable in situations where a constrained node interacts with a multitude of Internet hosts. In this case, a DTLS session has to be setup with every communicating host. Keoh et al. [7] have determined the communication overhead for a single-hop network without any packet loss to be an additional 12 messages spanning four extra round trips. These extra messages can be attributed entirely to the complex DTLS handshake.

There is also the problem of exchanging the necessary cryptographic information (e.g. a session key) between hosts. As constrained nodes often lack asymmetric cryptography support due to its high complexity, exchanging a session key can be challenging. This also means that a public key infrastructure - which is very popular on the Web today - is unfeasible for most constrained networks.

Finally, configuring and enforcing access rules to sensitive information is an other important feature that is lacking in the IETF IoT stack. Some parties might have access to all CoAP resources on a constrained device, while other should only have access to one or two resources. Implementing such an elaborate and fine-grained access control mechanism is too complex for constrained devices.

B. Limited amount of network resources

The end-to-end connectivity provided by the IETF stack enables direct interactions with constrained devices. While this allows applications to interface directly with constrained devices themselves, caution should be exercised in order to avoid unwanted depletion of the resources of the network and its devices. As an example, the owner of a constrained network will refrain from deploying the CoAP stack when it does not prevent abuse of the network by e.g. rapidly draining the network's energy reserve or computationally overloading specific devices in the network. Clearly, additional control of the network access is necessary to avoid such abuse by malicious users. Note that managing Internet access to the constrained network differs from secure communication (e.g. via DTLS), i.e. the former is almost always necessary while the latter is not.

C. Exposing additional information

The set of information offered by a network composed of class 1 devices is often determined at deploy time. This is because it is cumbersome to update these devices with new software once they are in the field. Therefor it is difficult to offer new information in the form of CoAP resources on devices that are already deployed. Moreover, it is not always possible to store this information on the device so that it can be recuperated after a device reset. In other cases, the information cannot be generated by the device (think of diagnostic networking and routing information) or the cost of transporting it to the device is too high. Note that this cost goes up with the volatility of the information.

III. Intercepting intermediary concept

All of the problems presented in the previous section are difficult to solve by focusing only on class 1 devices. There are two primary reasons for this, firstly a class 1 device lacks the resources necessary to tackle these issues. Secondly, some issues can not be solved by the device itself because they require e.g. a global overview of all interactions with the network or because the goal is to minimize the amount of interactions with the device. Therefor, our approach aims to solve these issues by relying on an intermediary party, that is less constrained than a class 1 device, to do the necessary "heavy lifting". In Internet integrated networks, this intermediary is usually the edge router or Internet gateway. Thus, their typical tasks like bridging link-layer technologies, network routing and firewalling are expanded with responsibilities at the transport and application layers.

We decided to pursue the concept of an intercepting intermediary. This is a network node that processes all passing traffic even though it is neither the source nor the destination of said traffic. By focusing on an intermediary node and its transparency towards end hosts, changes to existing infrastructure are kept to a minimum which makes our solution easy to deploy alongside existing IoT networks. For the best results, the intermediary should be deployed at the edge of the constrained network (e.g. at the Internet gateway). However, while not presented here, a distributed deployment with a cloud-computing component is also possible.

An overview of the intercepting intermediary is shown in Figure 2. Packets going to and from the constrained network are handled on the intermediary by an *Intercept* element, which decides based on information available at the matching service whether the intermediary should perform any processing on the CoAP message. The interception component (lower right of the figure) contains a lists of transformations that are to be applied for specific CoAP URIs on a per host or per subnet basis. These transformations are ordered lists of adapters, such a list is called an adapter chain. Adapters are function blocks that process CoAP messages. They can implement a wide range of functionality: amongst other things they can alter CoAP requests and the responses, inspect passing packets, consume external services and autonomously respond to requests on behalf of constrained devices.

The *Matching service* element is responsible for executing each adapter in the chain in sequence, while all the time passing along the intercepted CoAP message. When one of

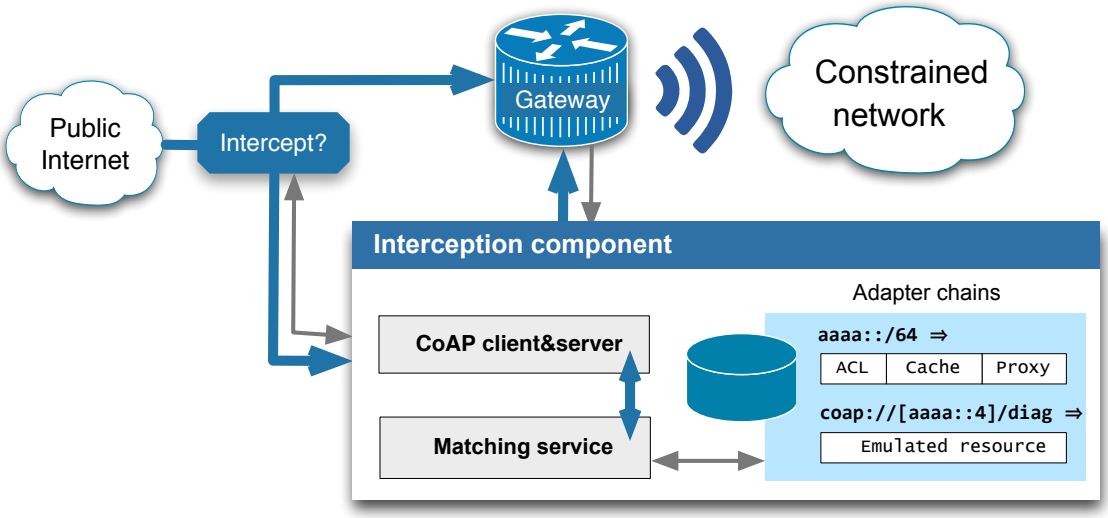

Fig. 2: Overview of intercepting intermediary for fine-grained management of CoAP interactions

the adapters signals that a response is available, execution of the message in the chain is stopped and the response traverses the chain in reverse allowing each adapter to alter the response. After traversing the adapter chain in reverse, the response is sent back to the source of the request. This process of messages and responses traversing through the adapter chain is illustrated in Figure 3.

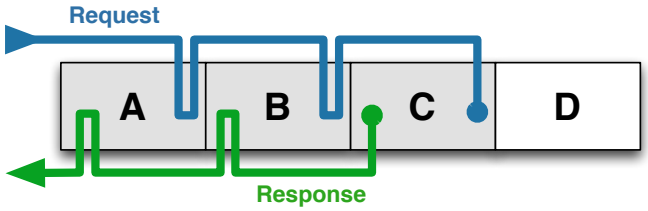

Fig. 3: Flow of a CoAP messages and responses through an adapter chain

The intercepting intermediary, adapter chains and matching filters are the key concepts that offer the necessary flexibility for reconfigurable, fine-grained and device-specific management and control.

IV. Intercepting Intermediary for Managing CoAP Interactions in Constrained Node Networks

Our approach for tackling the issues identified with deploying DTLS is two-fold and depends on whether the constrained device is capable of supporting the DTLS protocol considering its footprint. In both cases we suggest to terminate the DTLS session with the public Internet host at the gateway. For DTLS-capable devices, a separate long-lived session is maintained between the gateway and the device. For devices not capable of supporting DTLS, we suppose either using no security at all (thus DTLS is only applied on the public leg of the communication path) or relying on application-layer security, where all data above the transport layer is encrypted using a shared secret. The latter approach can be realized in a smaller footprint than DTLS with PSK cipher suites as e.g. the handshake mechanism can be omitted. This idea lies on the assumption that there is a trust relationship between the device and its gateway. Setting up a secure channel between the gateway and the device, allows to multiplex CoAP messages to and from a multitude of hosts over this channel. Note that terminating the DTLS session at the gateway is done in a transparent manner, from the point of view of the Internet host it appears as if it is communicating with the device directly over DTLS. No changes to the public Internet host are necessary. This approach allows to overcome the large footprint of DTLS on class 1 devices by relying on application layer security while still supporting DTLS to the public Internet. It also avoids the high communication overhead of having to setup ephemeral DTLS sessions for every public Internet host. Finally, cryptographic information only has to be exchanged once between the device and its gateway; which greatly eases key distribution.

For managing access to the limited amount of network resources three ideas are formulated in this paragraph. As CoAP has a built-in albeit simple congestion control mechanism, that strictly limits the amount of simultaneous outstanding interactions of a client to a server. By default, the limit is one connection. By monitoring traffic at the gateway for all hosts communicating with the constrained network, we can enforce this mechanism and filter out clients that are not following CoAP's congestion control mechanism. In case of a confirmable request, retransmissions can be delayed at the gateway when they arrive before the average round trip time to the constrained node has expired. Apart from end-to-end congestion control mechanisms, the gateway can also enforce more global policies where the rate of interactions with the entire constrained network can be limited. A straightforward but effective method to cope with the limited amount of network resources, is to employ a CoAP cache at the edge of the network. Not only does this improve response times considerably for cacheable CoAP resources in the network, it also decreases the amount of CoAP messages that have to processed by the constrained network. Our third and final suggestion for controlling CoAP interactions, is to apply an access control list at the edge of the network. When compared to an ACL on the device itself, unauthorized communication can already be dropped at the edge of the network without the

network having to waste resources in routing and processing messages to their destination where they would be dropped anyway.

For offering information that can not be stored at the constrained device we propose to emulate CoAP resources on behalf of the device at an intermediary. This has to the benefit of alleviating the device of storing and communicating the resource everytime it is accessed or updated. In order to make emulation of CoAP resources completely transparent to the outside world, the responses appear to originate from the device. Also, the emulated resources have to be injected in CoAP's .well-known/core discovery resource on the device itself.

V. Initial Implementation and Results

Our intercepting intermediary is built using Click, a modular software router [8]. The implementation expands upon previous work performed at our research group. The adapters allow for a modular approach to implementing our suggestions and at the time of writing these adapters are implemented:

(a) CoAP congestion control adapter: implements the default CoAP CC mechanism detailed in section IV.
(b) CoAP caching adapter: serves CoAP responses.
(c) CoAP resource emulation and .well-known/core rewriting adapters: emulates CoAP resources on behalf of constrained devices and makes these discoverable
(d) CoAP proxy adapter: this is usually the last adapter in a chain, it is used to fetch a response from the constrained network.

Note that implementation work on DTLS adapters is ongoing, but that it is currently too premature to be mentioned here.

Creating and configuring a chain of adapters on the intermediary is accomplished via a CoAP request. In the listing below a management client (typically the owner of the constrained network) executes a POST request to a subresource of adapterChains on the intermediary. The name of the subresource determines the set of constrained devices for which network traffic should be intercepted and passed on to the adapter chain. Apart from specifying entire IPv6 subnetworks, individual hosts can also be specified. The payload of the request contains an ordered list of adapters (the 'chain' key in the dictionary) and a path further specifying for which CoAP resources the chain has to be applied. In this example a chain that enforces CoAP's congestion control, caches CoAP responses and fetches responses from the network is created for all CoAP resources on all hosts in the aaaa::/64 subnet. The intermediary responds with the location of the CoAP resource that is created to allow future changes to the adapter chain. Parameters of specific adapters can also be configured at this location.

```
Management client                Intermediary
 | --  POST, /adapterChains/aaaa::˜64 --> |
 |      PAYLOAD: {"path": "*", "chain":   |
 |               ["congestioncontrol",    |
 |                "cache","proxy"]}       |
 |                                        |
 | <-- 2.01 Created                  --- |
 |         /adapterChains/aaaa::˜64       |
```

This adapter chain is used in an example when retrieving a CoAP resource of variable size from a constrained device. In a first scenario the chain is activated, while in the second scenario it is not. In both scenarios the number of packets transmitted by the intermediary (including retransmissions) and received by the constrained node are measured. The results for the two scenarios are shown in Figure 4. Note that mainly the influence of the cache is noticeable in this figure, however it also demonstrates that our approach of an intercepting intermediary is valid. Here the adapter chain is applied to an entire subnet, but it can also be applied to specific devices.

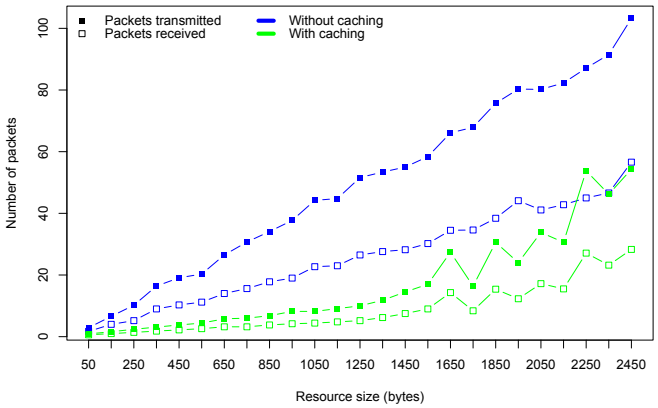

Fig. 4: Packets sent and received for increasing resource size

To illustrate that adapters can also be used to store and retrieve information for diagnostic purposes of the networks and its devices, consider a chain that consists of only a resource emulation adapter. The adapter implements a CoAP resource on behalf of the constrained device that combines operational information retrieved from the constrained node (e.g. remaining battery life, routing table) with information available at the gateway (e.g. number of forwarded bytes).

VI. Conclusion

This paper has listed a number of management issues commonly encountered when deploying the IETF IoT stack in constrained networks consisting of class 1 devices. Due to their unique characteristics, DTLS-based transport-layer security and fine-grained access control are challenging for this type of networks. Furthermore, unwanted abuse of limited network resources should be kept to a minimum. Finally, class 1 devices sometimes have to be instrumented after they are deployed in the field. Following an approach with function blocks deployed at an intercepting intermediary, gives us the flexibility needed to solve these issues. Preliminary work has begun on implementing and evaluating our suggestions. As a long-term result of this work, the authors hope to see an uptake in the adoption of the open IETF IoT standards by vendors. This would lead to a truly open Internet of Things much like the conventional Internet of today.

Acknowledgment

The authors would like to acknowledge that part of this research was supported by the COMACOD project. The iMinds COMACOD project is cofunded by iMinds (Interdisciplinary institute for Technology) a research institute founded by the Flemish Government. Companies and organizations involved in the project are Multicap, oneAccess, Track4C, Invenso and Trimble, with project support of IWT.

REFERENCES

[1] M. Zorzi, A. Gluhak, S. Lange, and A. Bassi, "From today's intranet of things to a future internet of things: a wireless-and mobility-related view," *Wireless Communications, IEEE*, vol. 17, no. 6, pp. 44–51, 2010.

[2] I. Ishaq, D. Carels, G. K. Teklemariam, J. Hoebeke, F. V. d. Abeele, E. D. Poorter, I. Moerman, and P. Demeester, "Ietf standardization in the field of the internet of things (iot): A survey," *Journal of Sensor and Actuator Networks*, vol. 2, no. 2, pp. 235–287, 2013. [Online]. Available: http://www.mdpi.com/2224-2708/2/2/235

[3] C. Bormann, M. Ersue, and A. Keranen. (2014, Jan.) Terminology for constrained node networks. [Online]. Available: http://tools.ietf.org/html/draft-ietf-lwig-terminology-06

[4] C. Bormann, A. P. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes," *Internet Computing, IEEE*, vol. 16, no. 2, pp. 62–67, 2012.

[5] D. Guinard, "Towards the web of things: Web mashups for embedded devices," in *In MEM 2009 in Proceedings of WWW 2009. ACM*, 2009.

[6] Open Mobile Alliance. (2014, Jan.) Machine to machine (m2m) solutions — oma. [Online]. Available: http://openmobilealliance.org/about-oma/work-program/m2m-enablers/

[7] O. Garcia-Morchon, S. Kumar, and S. Keoh. (2014) Securing the ip-based internet of things with dtls. [Online]. Available: http://tools.ietf.org/html/draft-keoh-lwig-dtls-iot

[8] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, Aug. 2000. [Online]. Available: http://doi.acm.org/10.1145/354871.354874